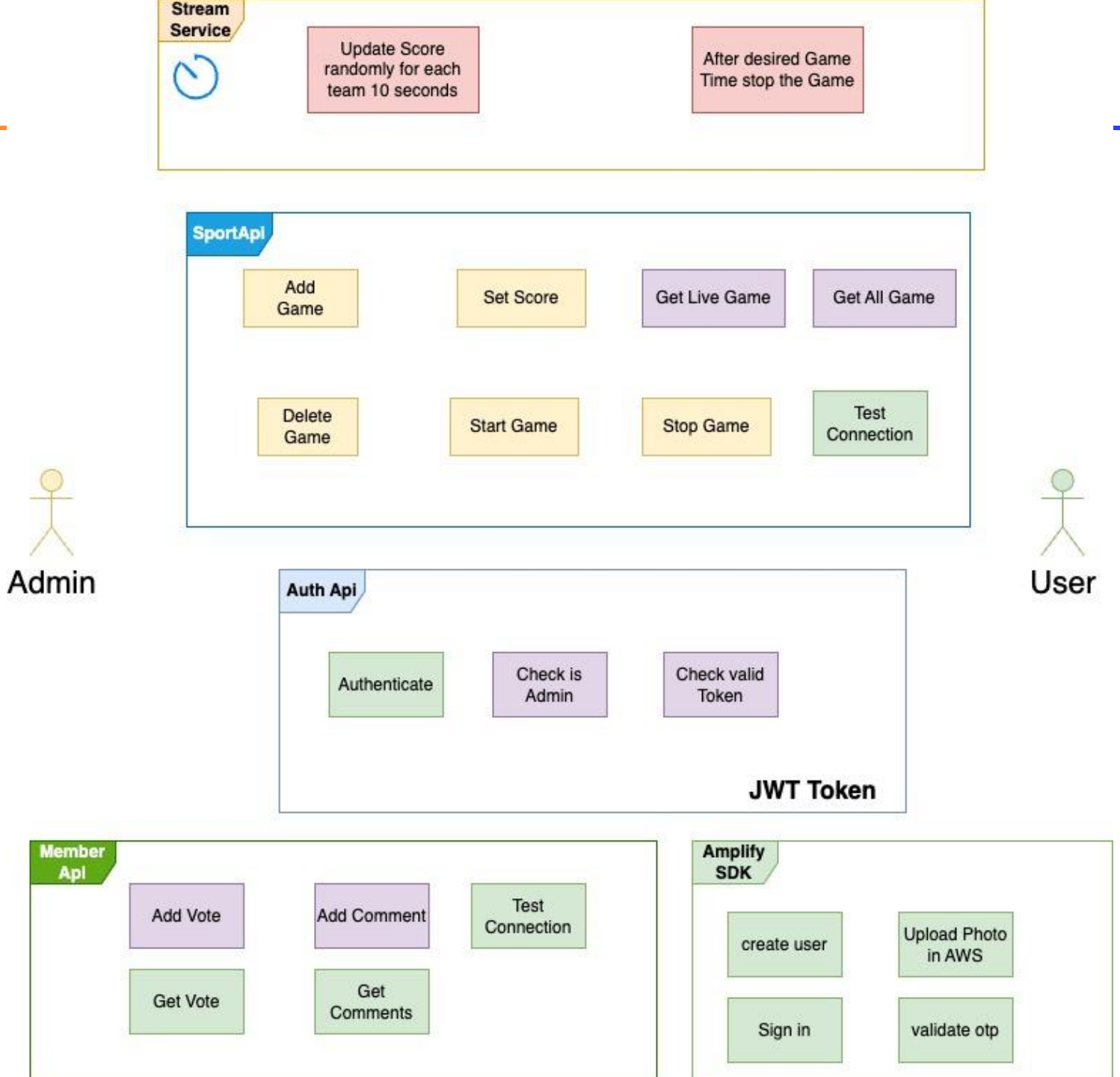
CS516PROJECT

Team Members

Ariunzaya Munkhbat
Bayarbayasagalan Jagdal
Syed Shujat Ali Shah
Thanh Do Nguyen
Duyen Tran



APPLICATION IDEA

Soccer Game

AWS Cloud

Microservices

Databases
(MongoDB + RDS)

S3 Storage
(Cloud-watch, images, logs)

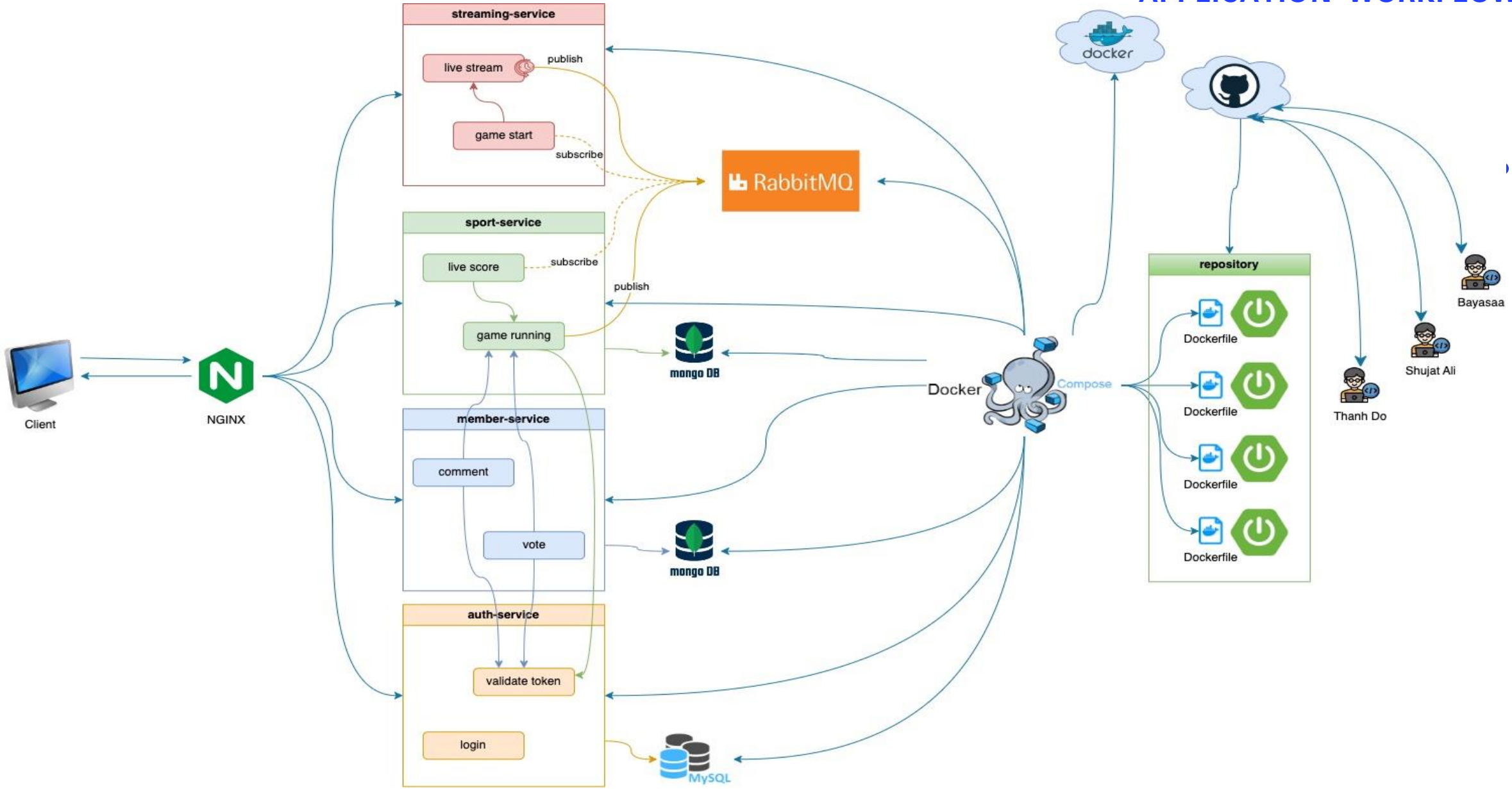
Lamba functions

Api Gateway

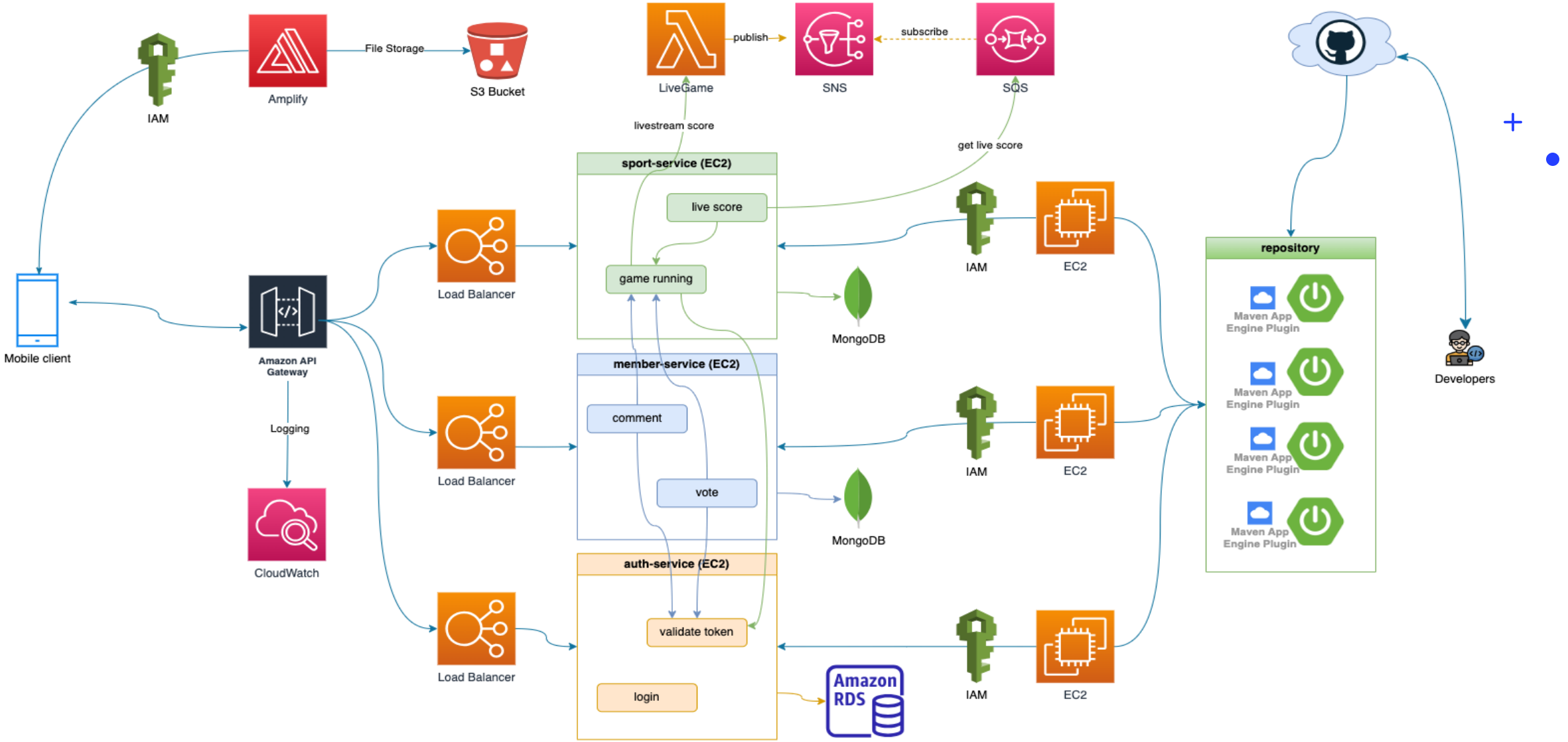
Android Mobile App Client

AWS Amplify

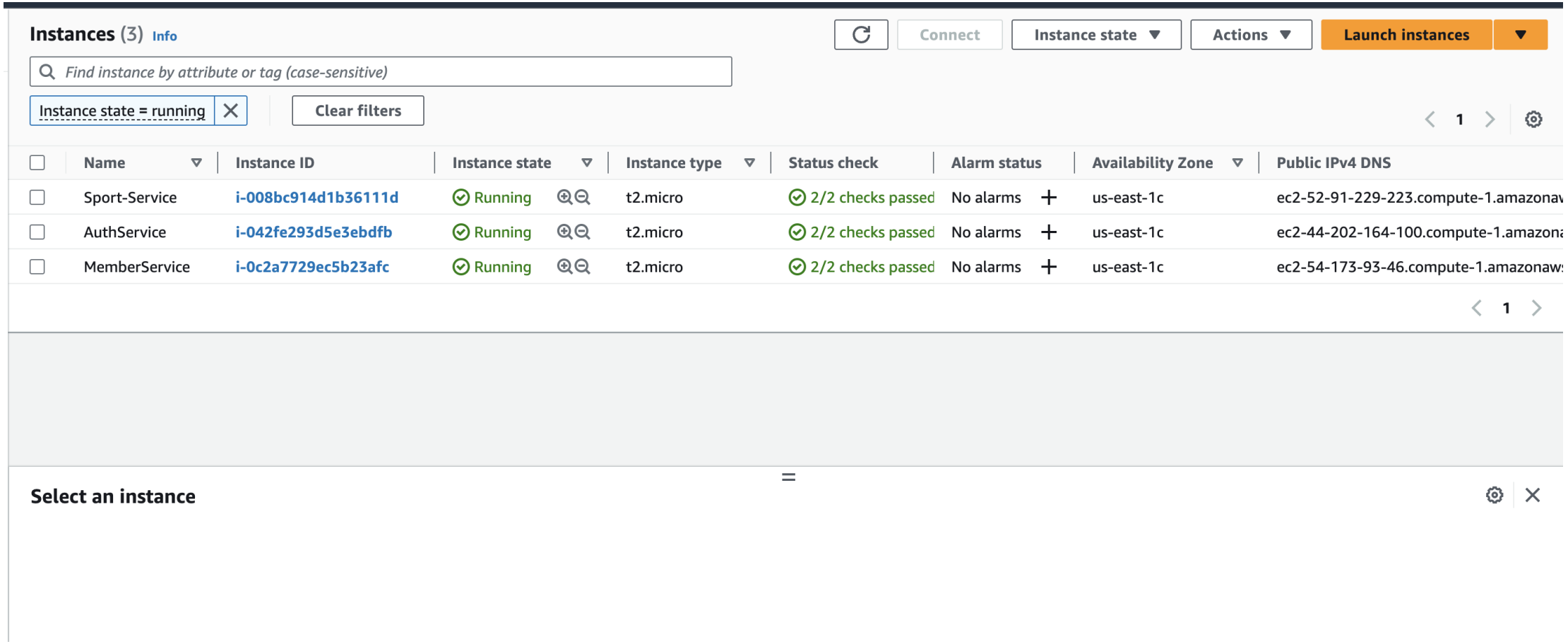
APPLICATION WORKFLOW



APPLICATION ON AWS



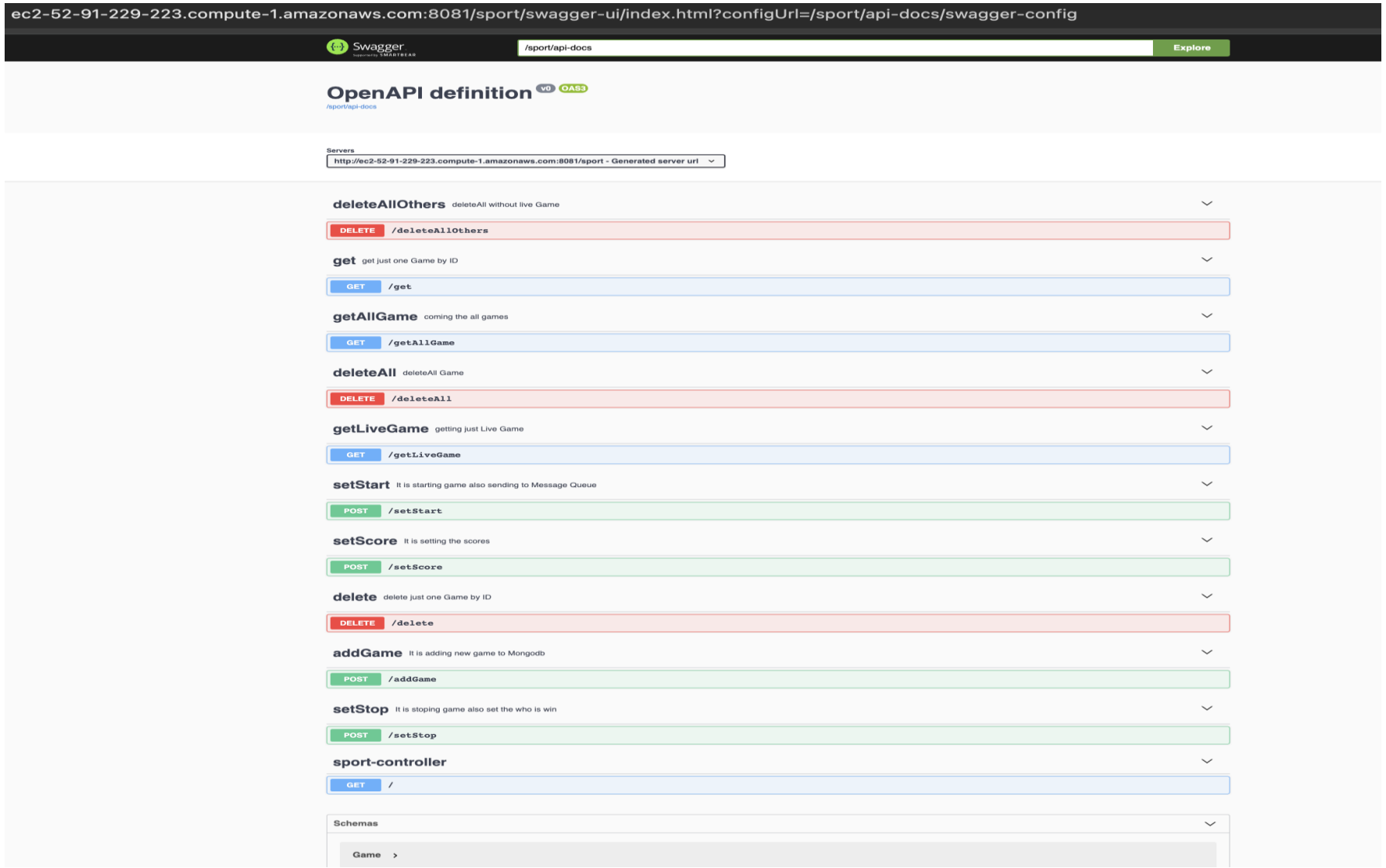
EC2 Instances



Running applications

```
git clone https://github.com/bayarbayasgalanj/cloud_computing.git
Install Java
sudo apt-get update
sudo apt install openjdk-17-jre
sudo apt install openjdk-17-jdk
sudo apt install  -y maven


Install Mongo
wget -qO - https://www.mongodb.org/static/pgp/server-4.2.asc | sudo apt-key add -
echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.2.list
sudo apt-get update
sudo apt install -y mongodb-org

echo "deb http://security.ubuntu.com/ubuntu focal-security main" | sudo tee /etc/apt/sources.list.d/focal-security.list

sudo apt-get update
sudo apt-get install libssl1.1
sudo apt install -y mongodb-org

sudo nano /etc/mongod.conf
net:
  port: 27017
  bindIp: 0.0.0.0

sudo systemctl start mongod
sudo systemctl enable mongod

Start service
mvn clean -f pom.xml
mvn package -f pom.xml
java -jar target/sport-service-1.0-SNAPSHOT.jar
```

EC2 Sport service



Lambda invoker from Sport service using AWS SDK

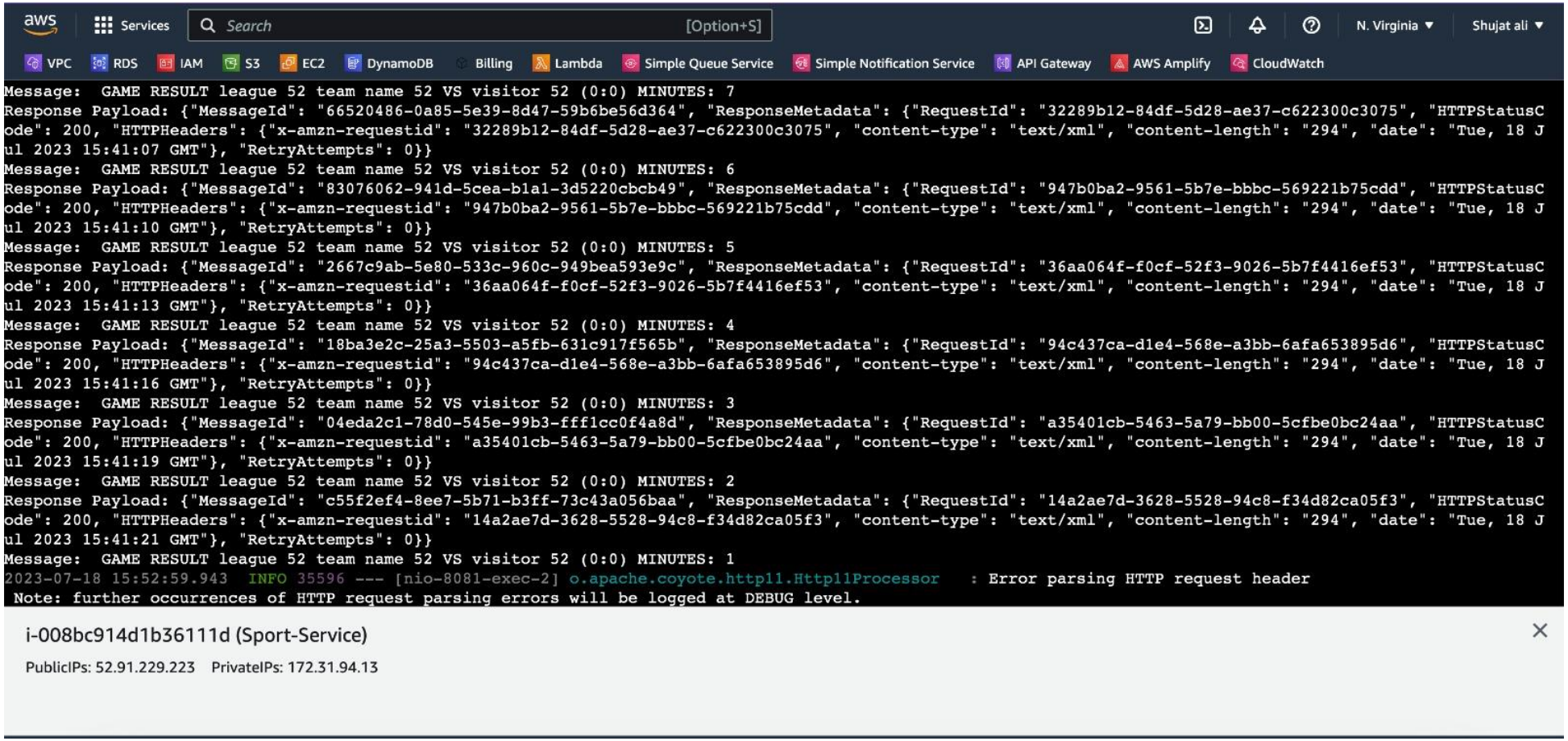
```java
nthanhdo2610
@Service
public class LambdaInvoker {

    @Value("${aws.accessKeyId}")
    private String accessKeyId;
    @Value("${aws.secretAccessKey}")
    private String secretAccessKey;
    @Value("${aws.lambda-function}")
    private String lambdaFunctionName;
    2 usages
    private LambdaClient lambdaClient;
    nthanhdo2610
    @PostConstruct
    private void init() {
        // Set up the Lambda client
        lambdaClient = LambdaClient.builder()
                .region(Region.US_EAST_1)
                .credentialsProvider(StaticCredentialsProvider.create(AwsBasicCredentials.create(accessKeyId, secretAccessKey)))
                .build();
    }

    3 usages  nthanhdo2610
    public void sendMessage(String message){
        // Create an InvokeRequest
        InvokeRequest request = InvokeRequest.builder()
                .functionName(lambdaFunctionName)
                .payload(SdkBytes.fromUtf8String("{\"message\":\" " + message + "\"}"))
                .build();

        try {
            // Invoke the Lambda function
            InvokeResponse response = lambdaClient.invoke(request);

            // Get the response payload
            String payload = new String(response.payload().asByteArray(), StandardCharsets.UTF_8);
            System.out.println("Response Payload: " + payload);
        } catch (LambdaException e) {
            System.err.println("Error invoking Lambda function: " + e.getMessage());
        }
    }
}
```

SQS Listener from Sport service using AWS SDK

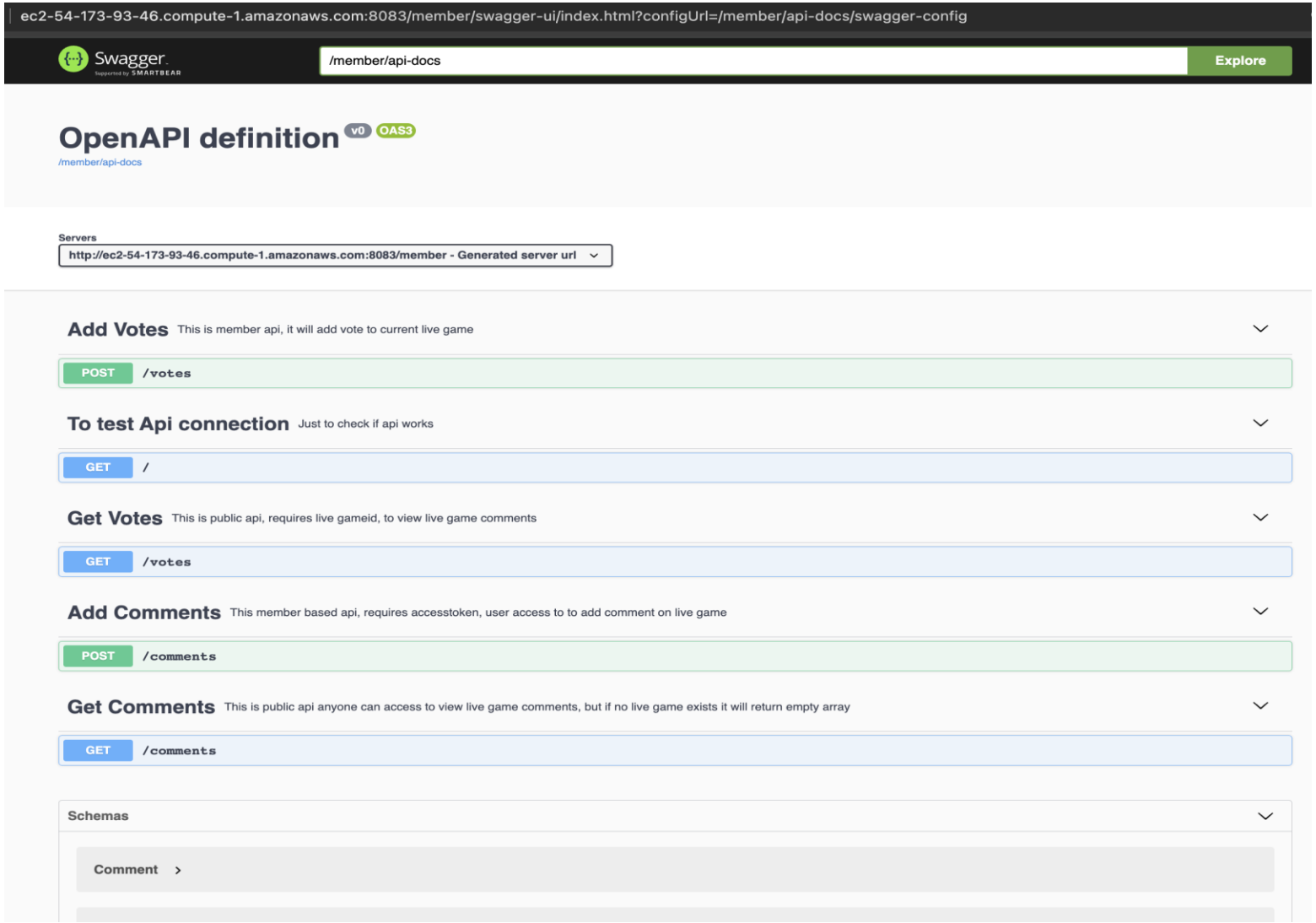
```java
public class SQSMessageListener {
    2 usages
    private final Region region = Region.US_EAST_1;
    @Value("${aws.accessKeyId}")
    private String accessKeyId;
    @Value("${aws.secretAccessKey}")
    private String secretAccessKey;
    6 usages
    private SqsClient sqsClient;
    5 usages
    private static String queueUrl;
    2 usages
    private ReceiveMessageRequest receiveRequest;
    nthanhdo2610 *
    @PostConstruct
    private void init() {
        SnsClient snsClient = SnsClient.builder().region(region)
                .credentialsProvider(StaticCredentialsProvider.create(AwsBasicCredentials.create(accessKeyId, secretAccessKey))).build();
        sqsClient = SqsClient.builder().region(region)
                .credentialsProvider(StaticCredentialsProvider.create(AwsBasicCredentials.create(accessKeyId, secretAccessKey))).build();
        // Create an SNS topic
        CreateTopicRequest createTopicRequest = CreateTopicRequest.builder().name( s: "LiveScore").build();
        CreateTopicResponse createTopicResponse = snsClient.createTopic(createTopicRequest);
        String topicArn = createTopicResponse.topicArn();
        // Create two SQS queues
        CreateQueueRequest createQueueRequest = CreateQueueRequest.builder().queueName( s: "Score").build();
        CreateQueueResponse createQueueResponse = sqsClient.createQueue(createQueueRequest);
        queueUrl = createQueueResponse.queueUrl();
        // Get the ARNs of the queues
        String queueArn = sqsClient
                .getQueueAttributes(GetQueueAttributesRequest.builder().queueUrl(queueUrl).attributeNames(QueueAttributeName.QUEUE_ARN).build())
                .attributes().get(QueueAttributeName.QUEUE_ARN);
        // Subscribe the queues to the SNS topic
        SubscribeRequest subscribeRequest1 = SubscribeRequest.builder().topicArn(topicArn).protocol( s: "sqs").endpoint(queueArn).build();
        snsClient.subscribe(subscribeRequest1);
        // Set the SQS queue attributes to receive messages from SNS
        Map<QueueAttributeName, String> queueAttributes = new HashMap<>();
        queueAttributes.put(QueueAttributeName.RECEIVE_MESSAGE_WAIT_TIME_SECONDS, "20");
        queueAttributes.put(QueueAttributeName.VISIBILITY_TIMEOUT, "300");
        SetQueueAttributesRequest setQueueAttributesRequest = SetQueueAttributesRequest.builder().queueUrl(queueUrl).attributes(queueAttributes).build(
        sqsClient.setQueueAttributes(setQueueAttributesRequest);
        receiveRequest = ReceiveMessageRequest.builder().queueUrl(queueUrl).maxNumberOfMessages( integer: 10).waitTimeSeconds( integer: 20).build();
    }
```

```java
nthanhdo2610 *
@Scheduled(fixedDelay = 1000)
public void receiveAndProcessMessages() {
    ReceiveMessageResponse receiveResponse = sqsClient.receiveMessage(receiveRequest);
    for (Message message : receiveResponse.messages()) {
        try {
            JSONObject obj = new JSONObject(message.body());
            System.out.println("Message: " + obj.getString( name: "Message"));
        } catch (JSONException e) {
            System.out.println("JSONException Message: " + e.getMessage());
        }
        deleteMessage(queueUrl, message.receiptHandle());
    }
}

1 usage  nthanhdo2610
private void deleteMessage(String queueUrl, String receiptHandle) {
    DeleteMessageRequest deleteRequest = DeleteMessageRequest.builder()
            .queueUrl(queueUrl)
            .receiptHandle(receiptHandle)
            .build();

    sqsClient.deleteMessage(deleteRequest);
}
```

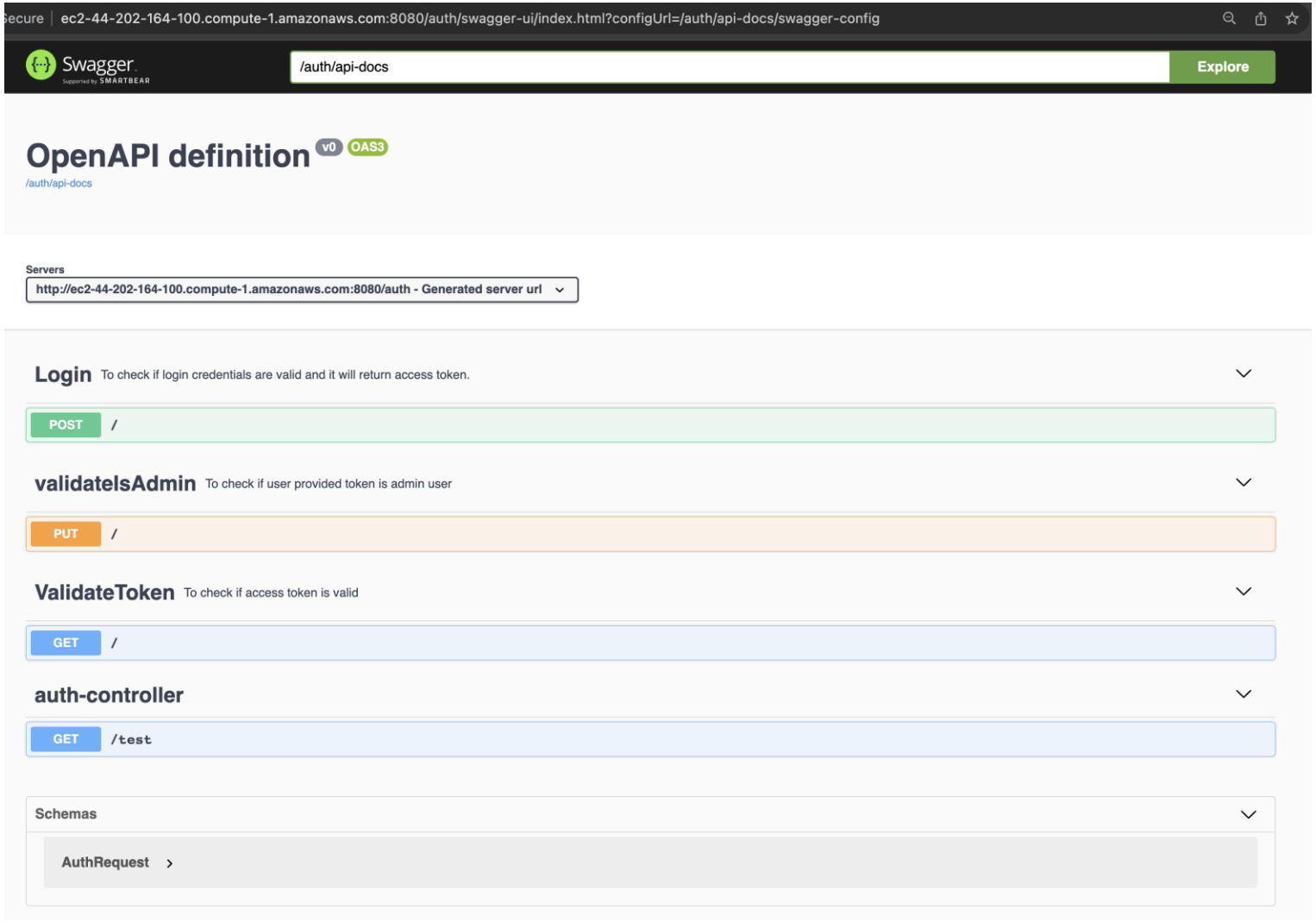
SQS Listener from Sport service using AWS SDK



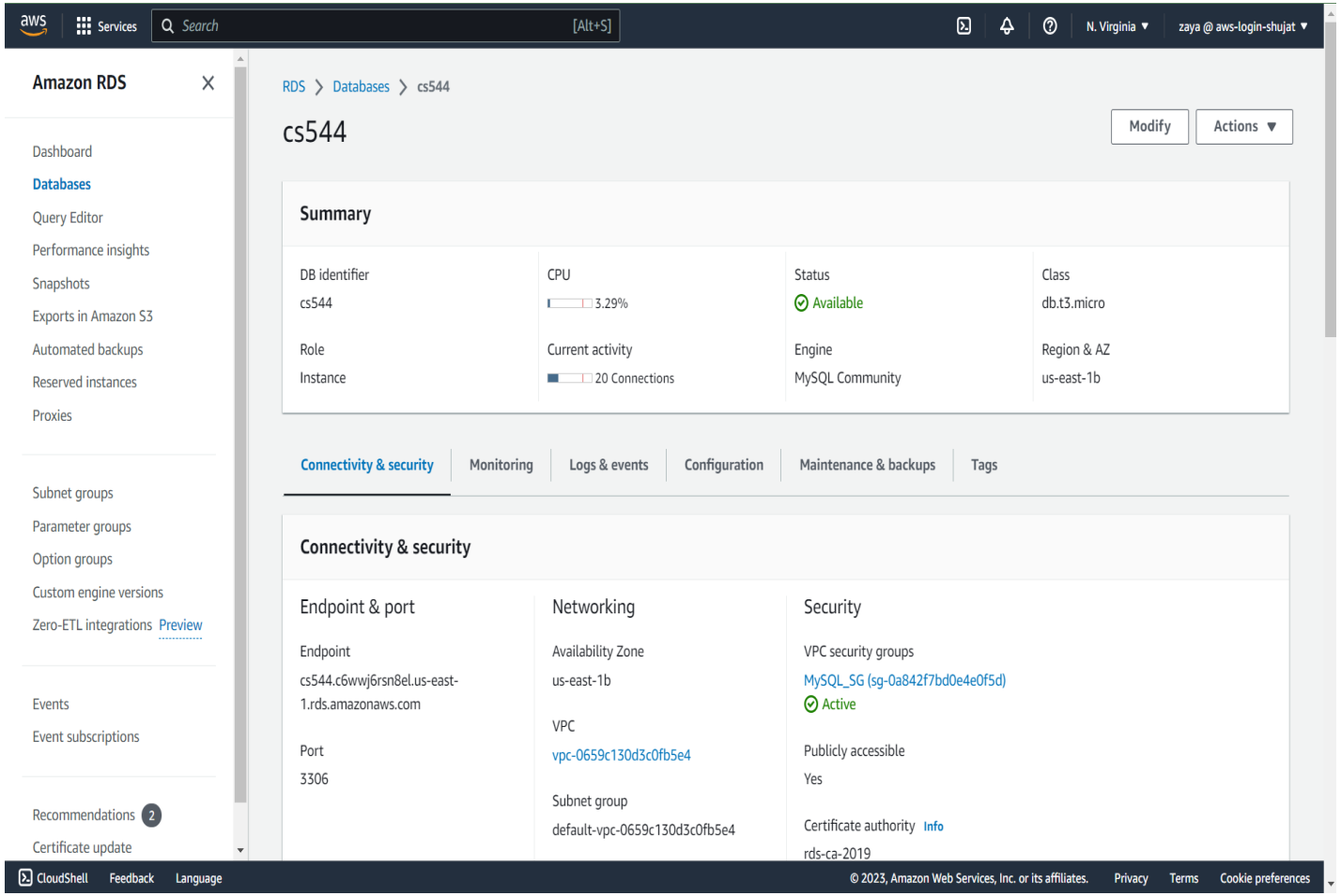
EC2 Member service



EC2 Auth service



Auth service - RDS



User Authentication:

Auth enables users to securely authenticate and access the system.
It supports various authentication methods, such as email and password.

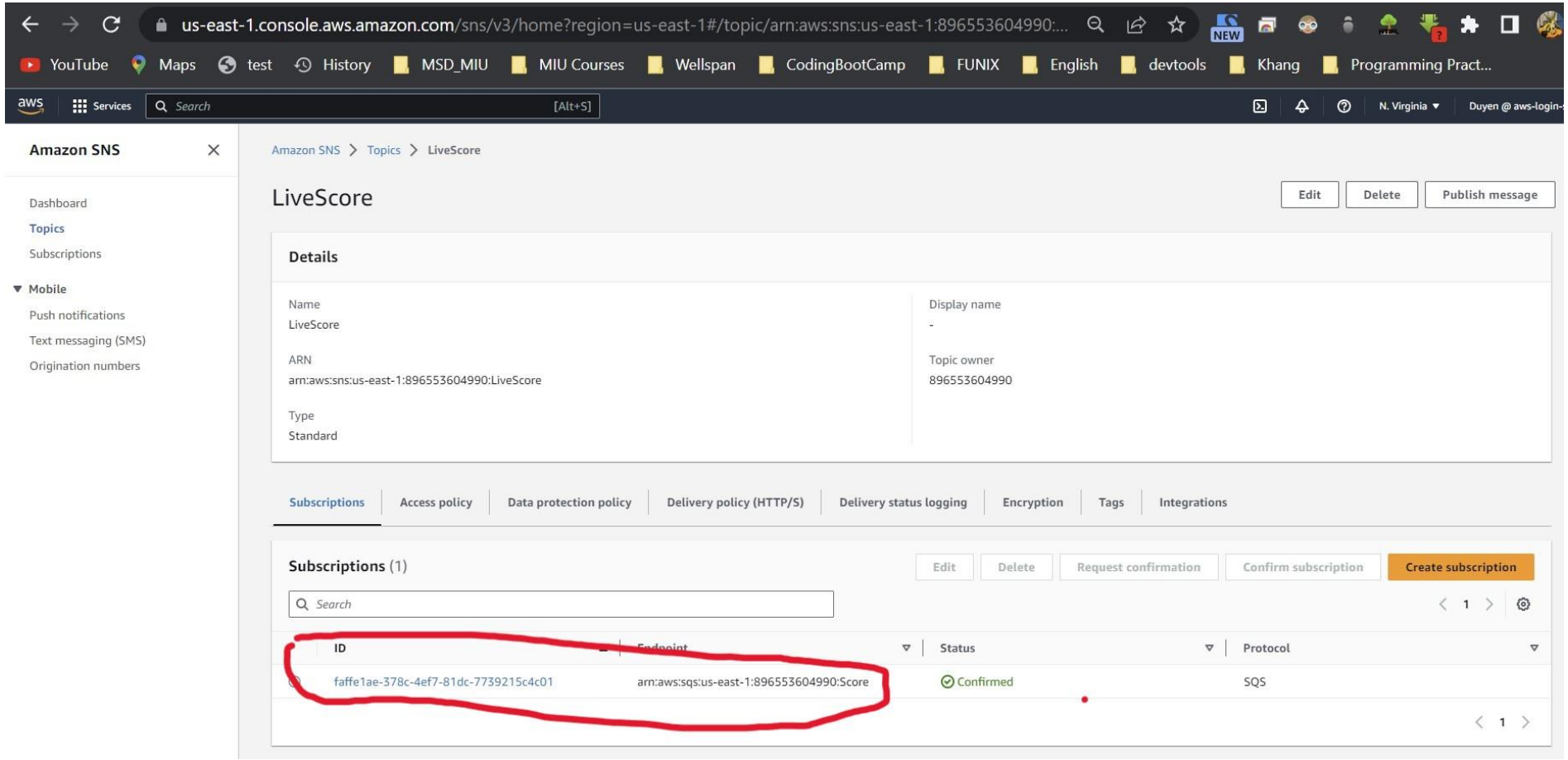
Token-based Access:

The service generates and validates access tokens using JSON Web Tokens (JWT).
This ensures secure and stateless communication between clients and the server.

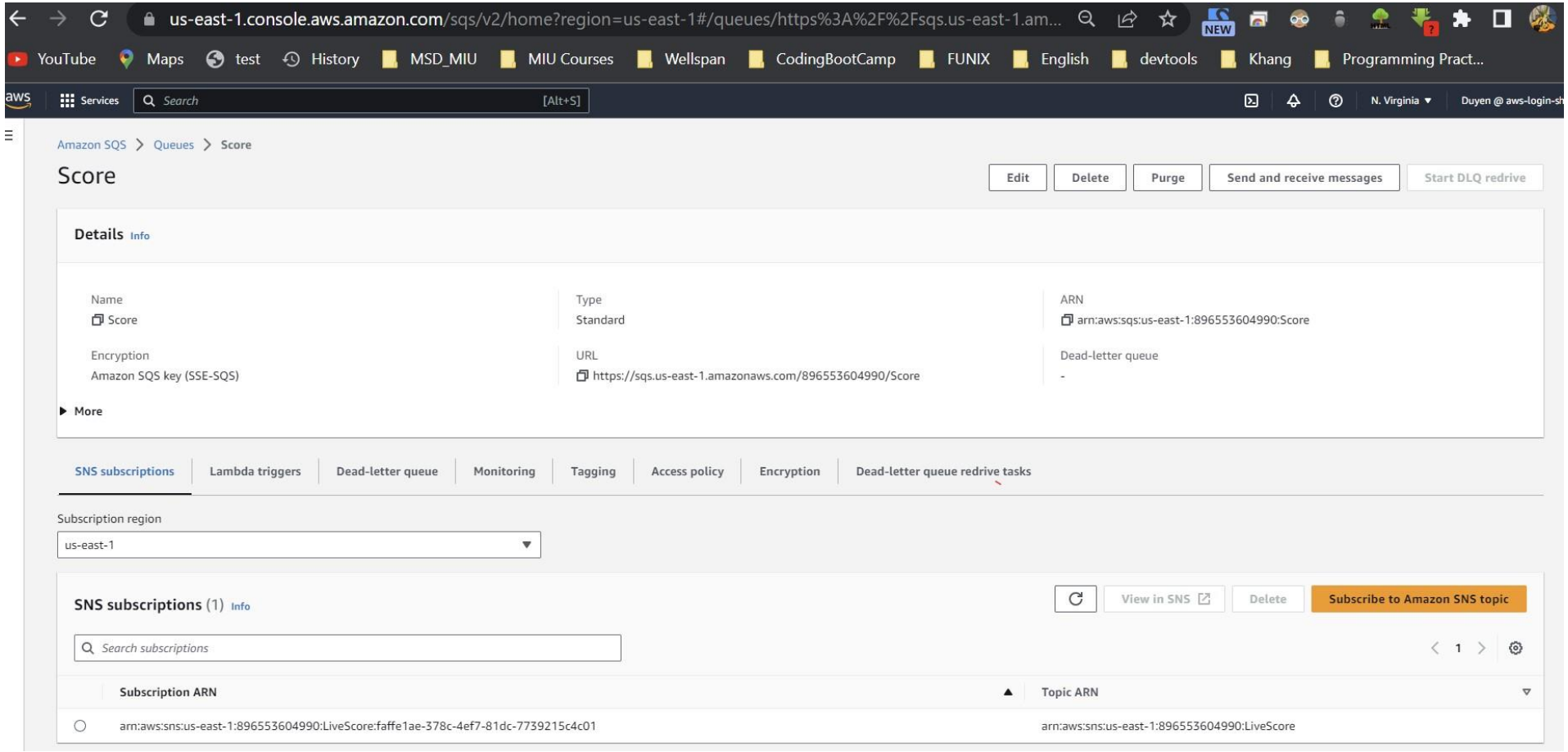
Admin Role:

Auth includes an admin role that grants additional privileges.
Admin users can manage system configurations, user roles, and permissions.

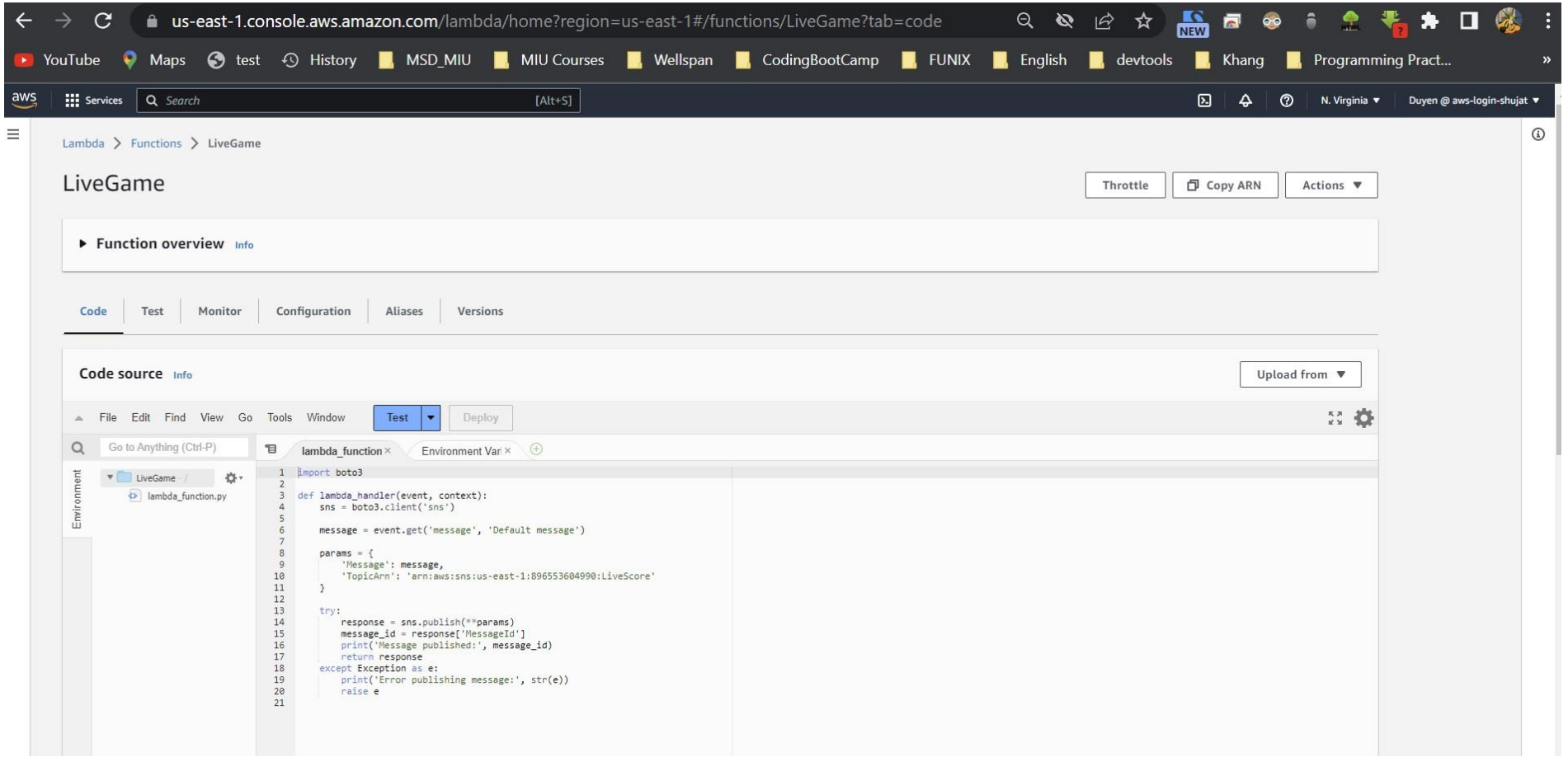
Creating SNS Topic



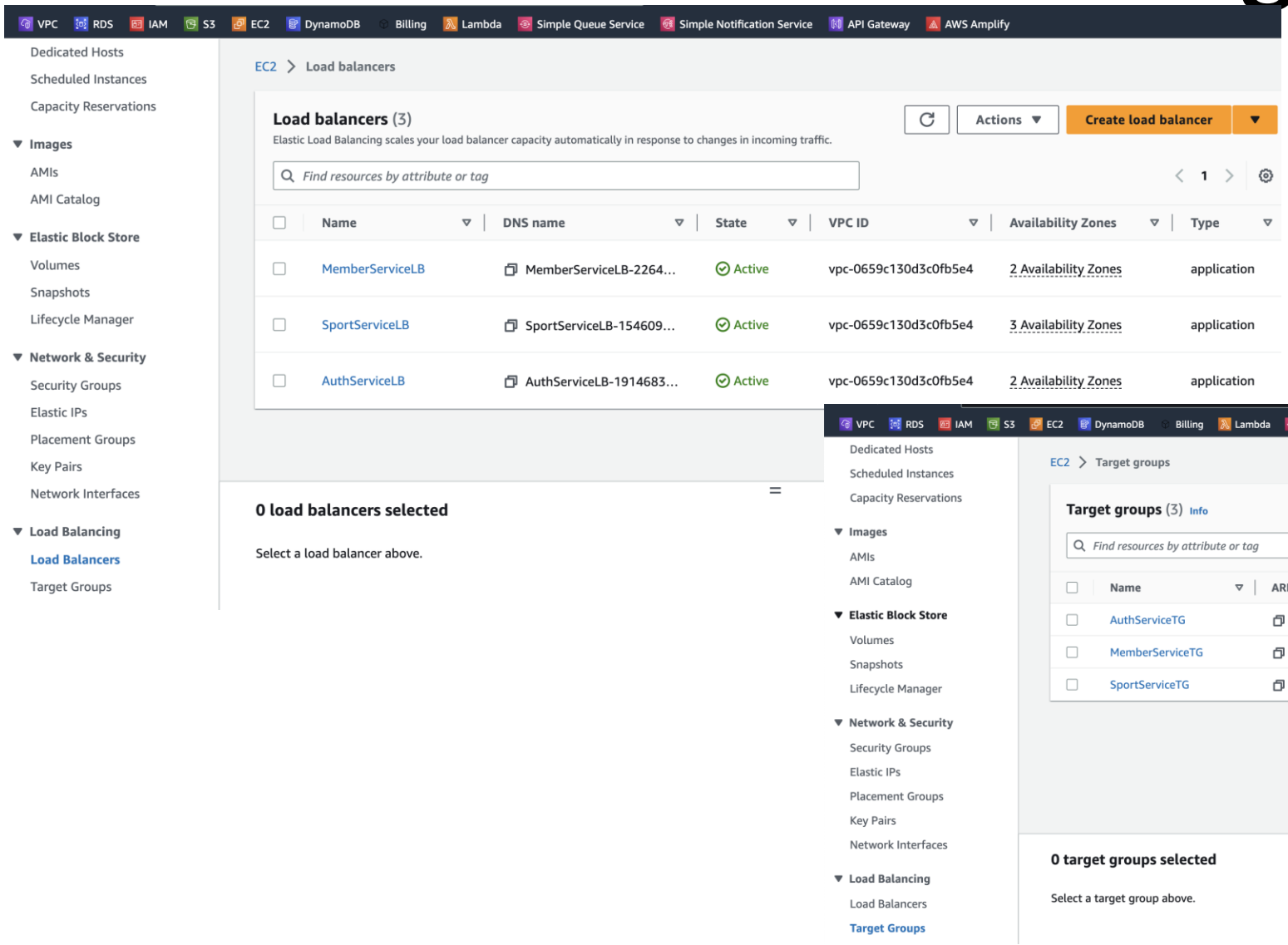
Creating SQS Queue

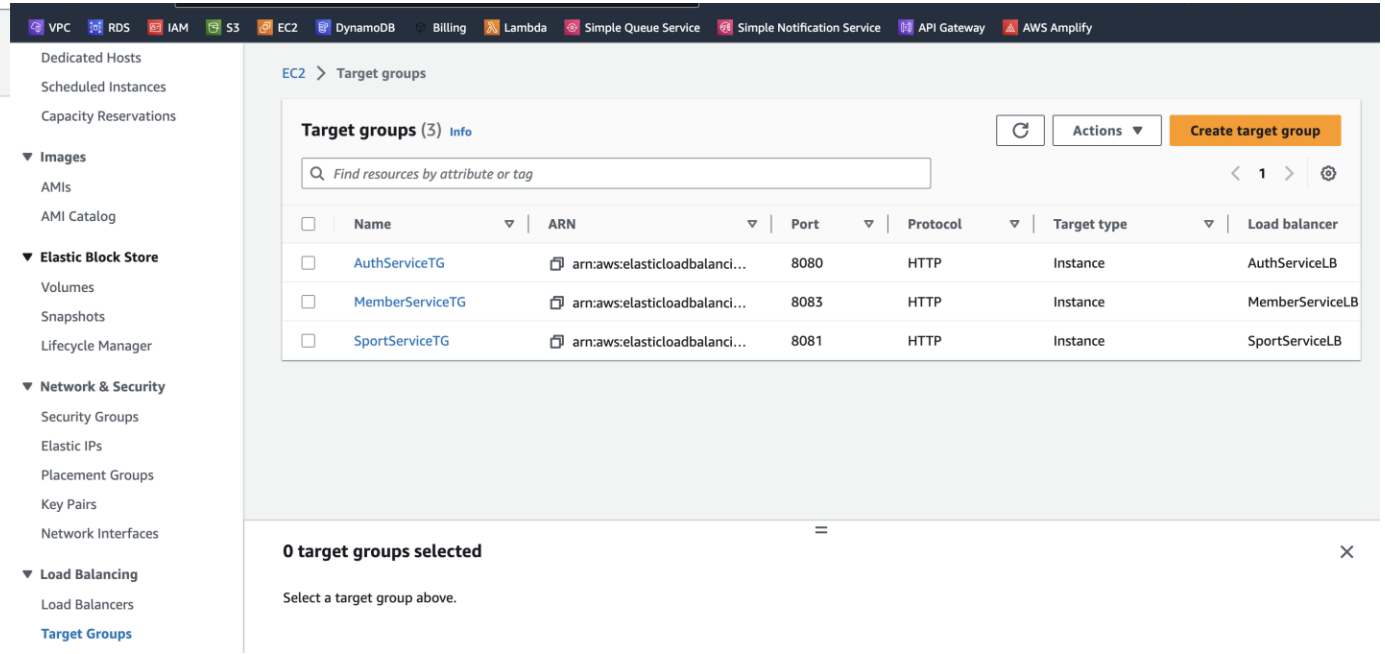


Creating Lambda Function

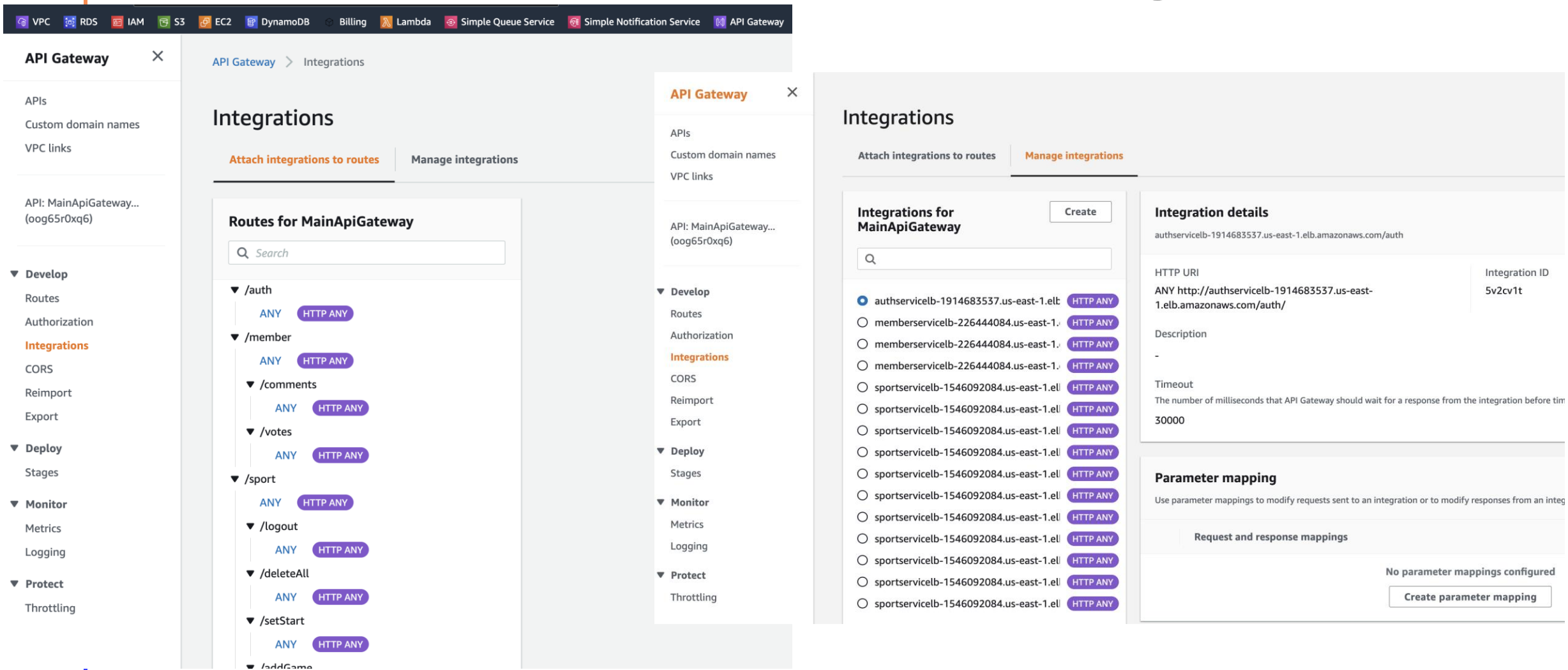


Load Balancers & Target Group

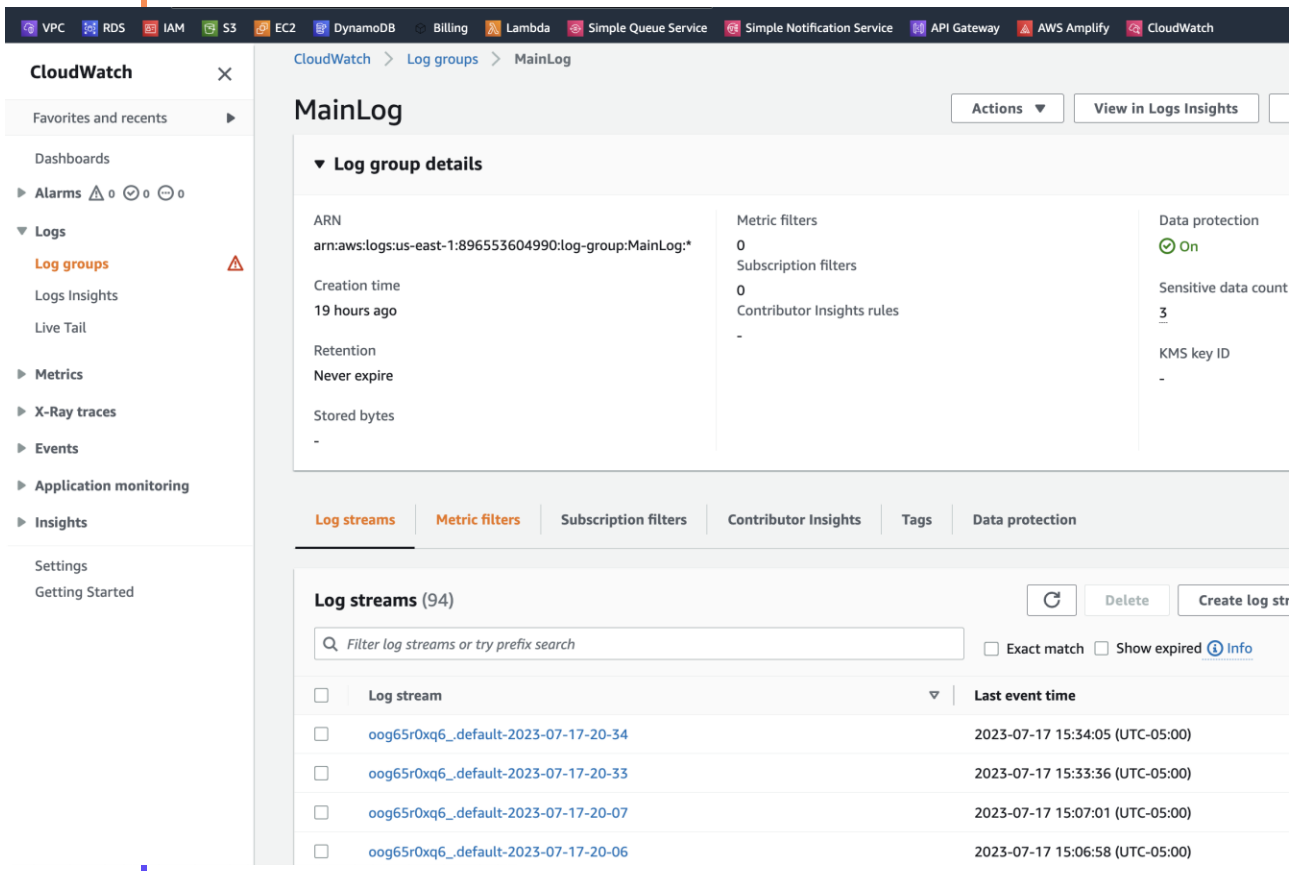


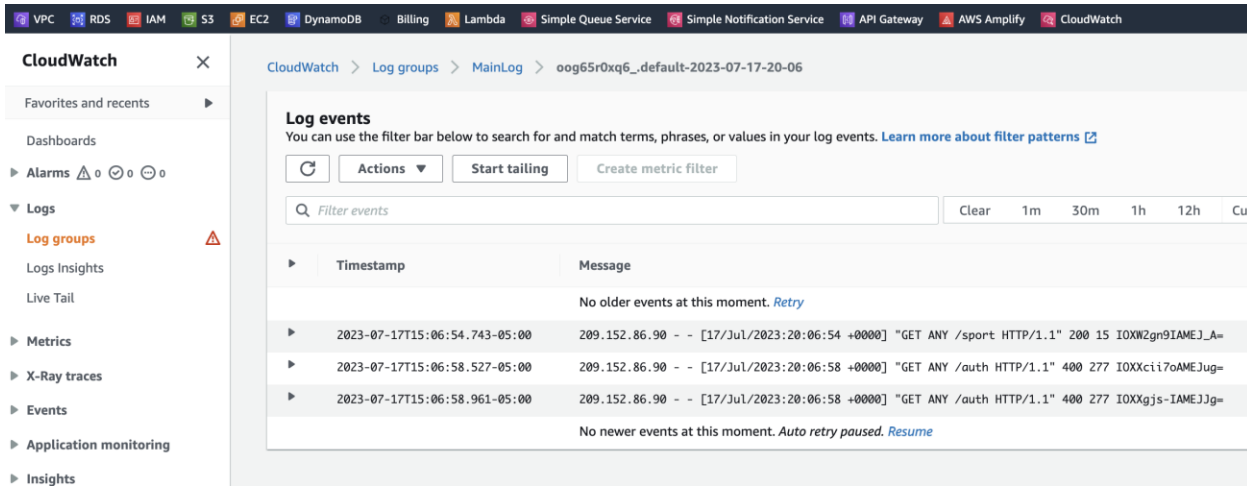


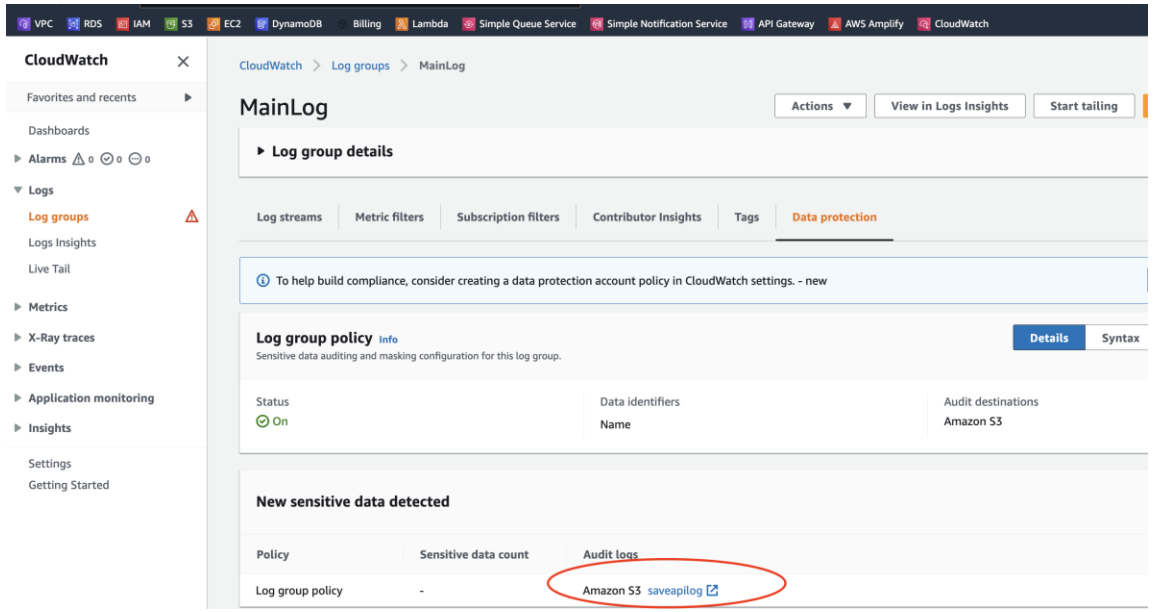
Api Gateway & Api Integrations



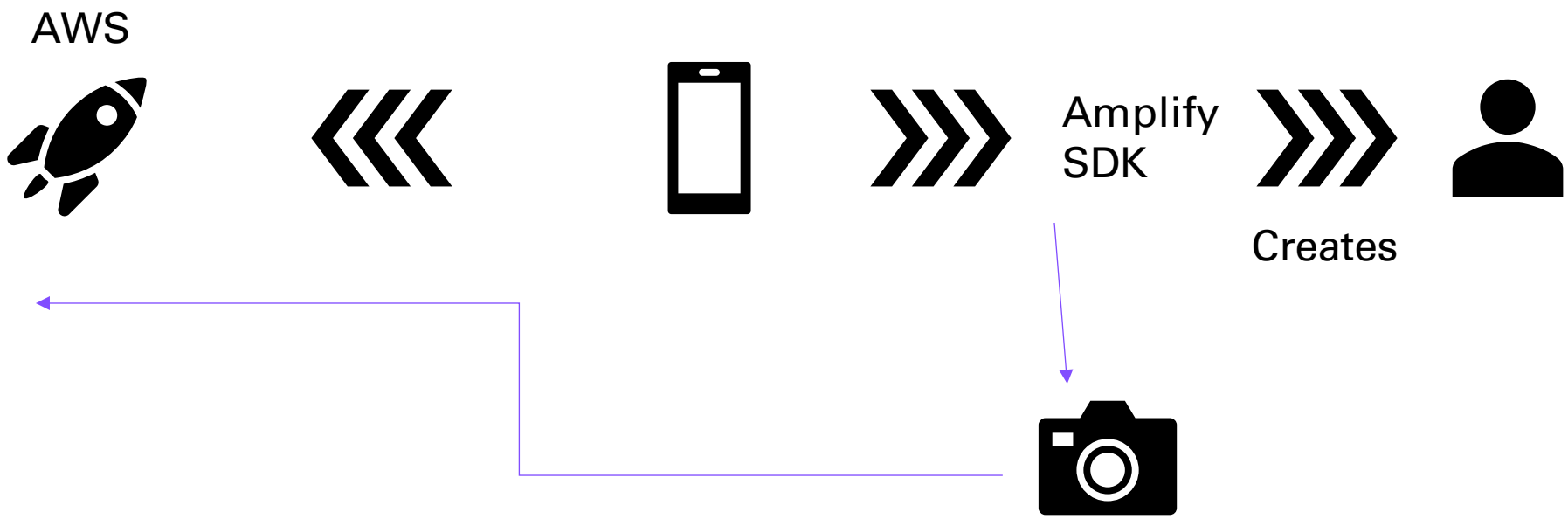
CloudWatch & Logging S3



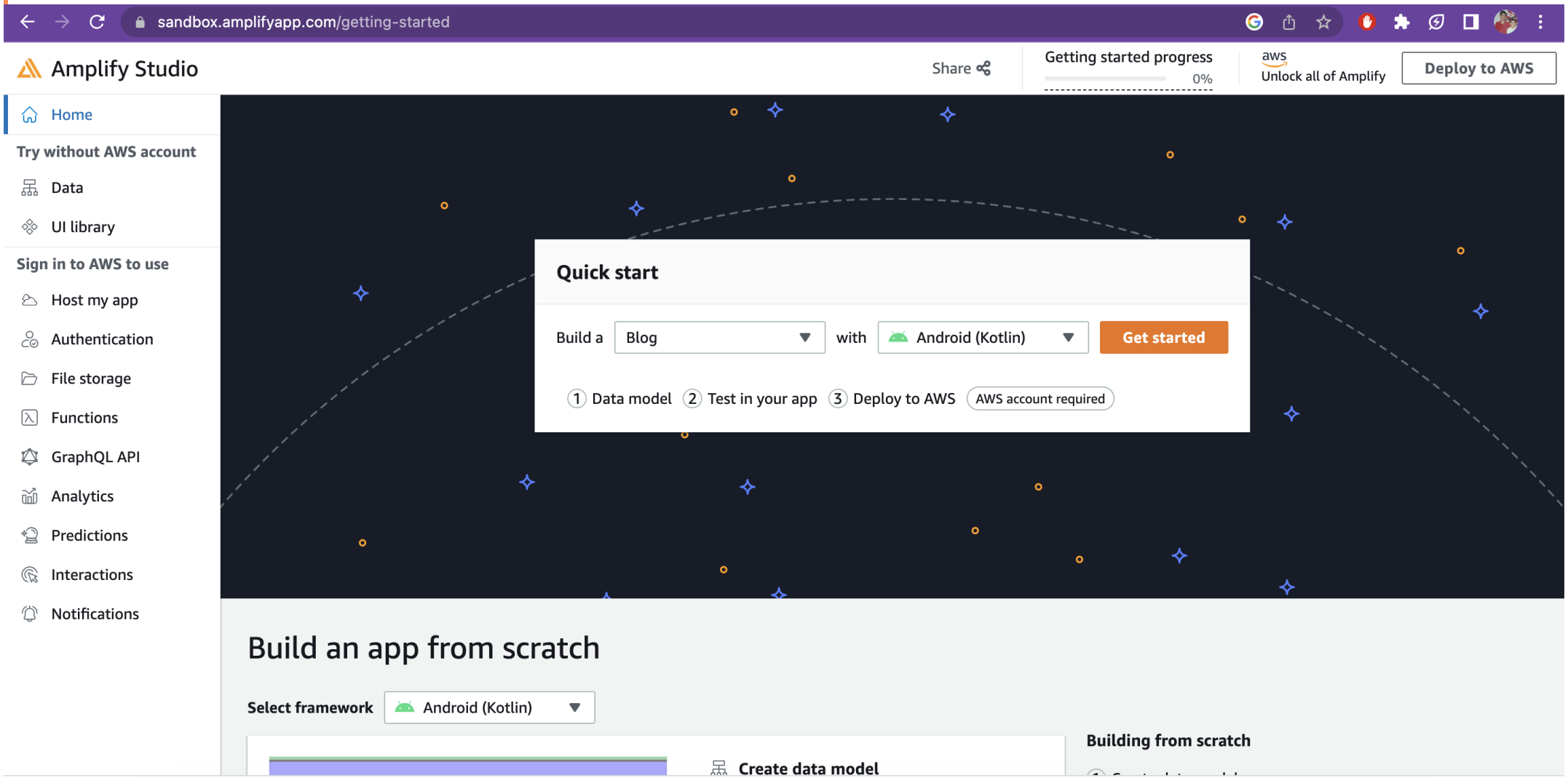




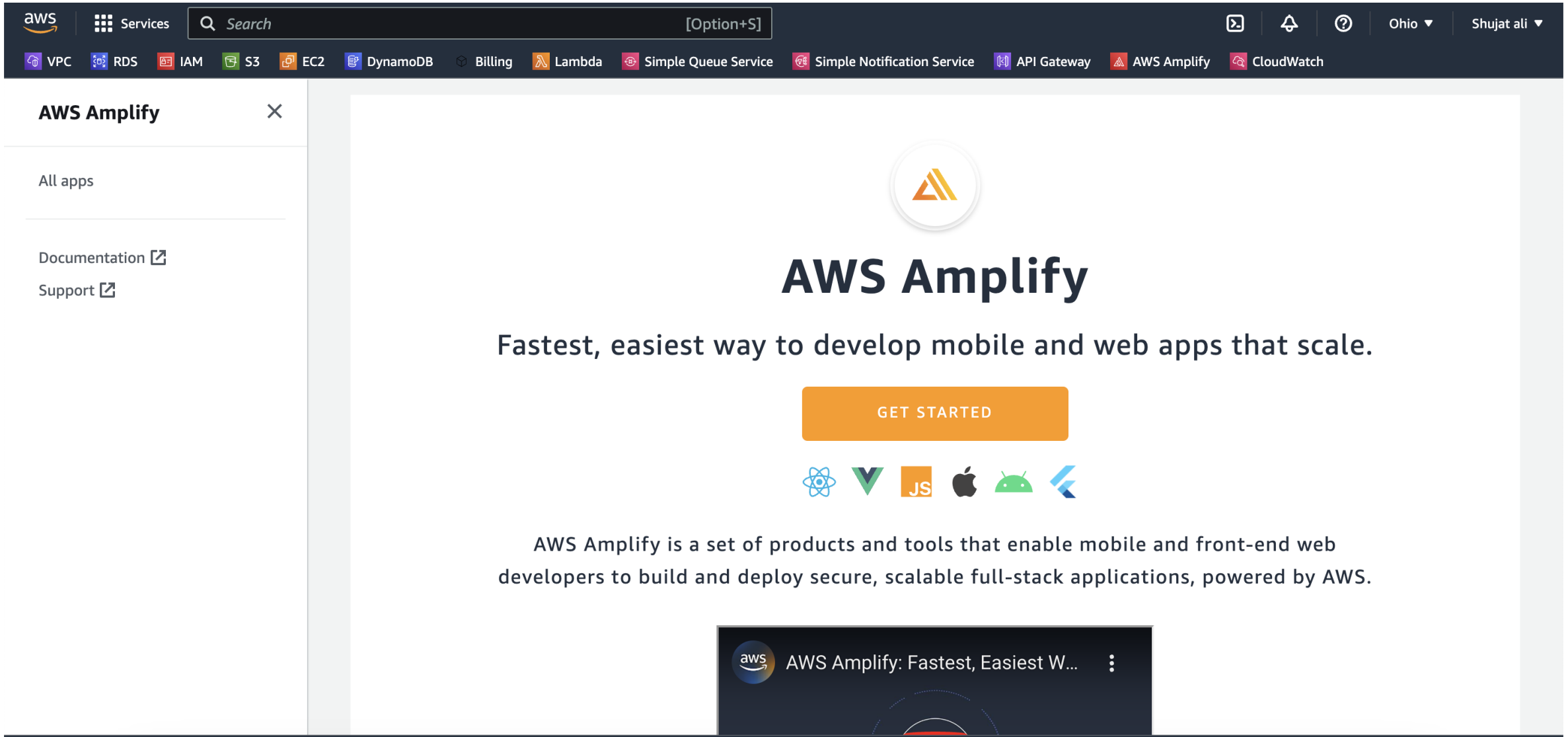
AWS Amplify And Mobile



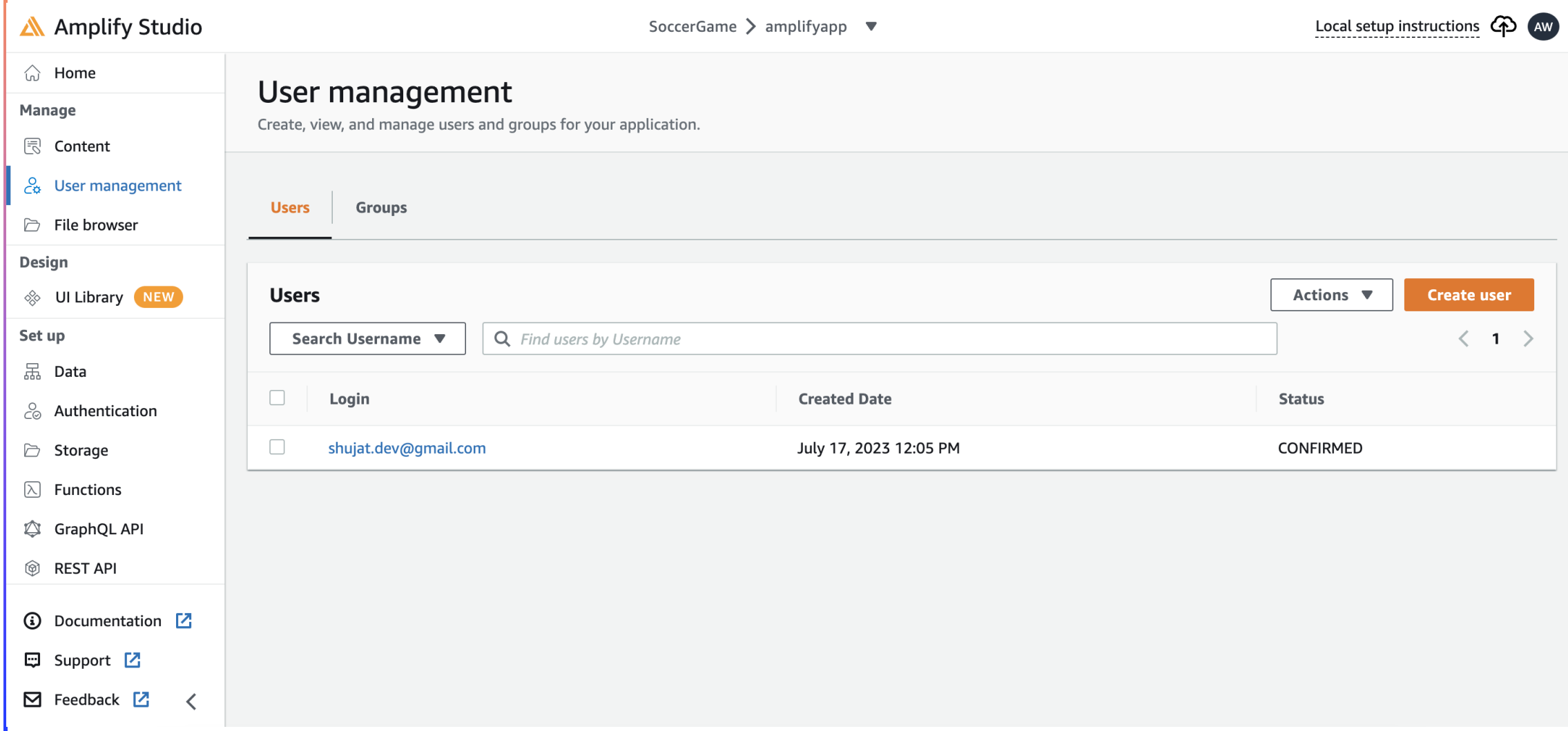
AWS Amplify


sandbox.amplifyapp.com/getting-started
Amplify Studio
Share
Getting started progress
0%
aws
Unlock all of Amplify
Deploy to AWS
Home
Try without AWS account
Data
UI library
Sign in to AWS to use
Host my app
Authentication
File storage
Functions
GraphQL API
Analytics
Predictions
Interactions
Notifications
Quick start
Build a
Blog
with
Android (Kotlin)
Get started
1 Data model
2 Test in your app
3 Deploy to AWS
AWS account required
Build an app from scratch
Select framework
Android (Kotlin)
Create data model
Building from scratch

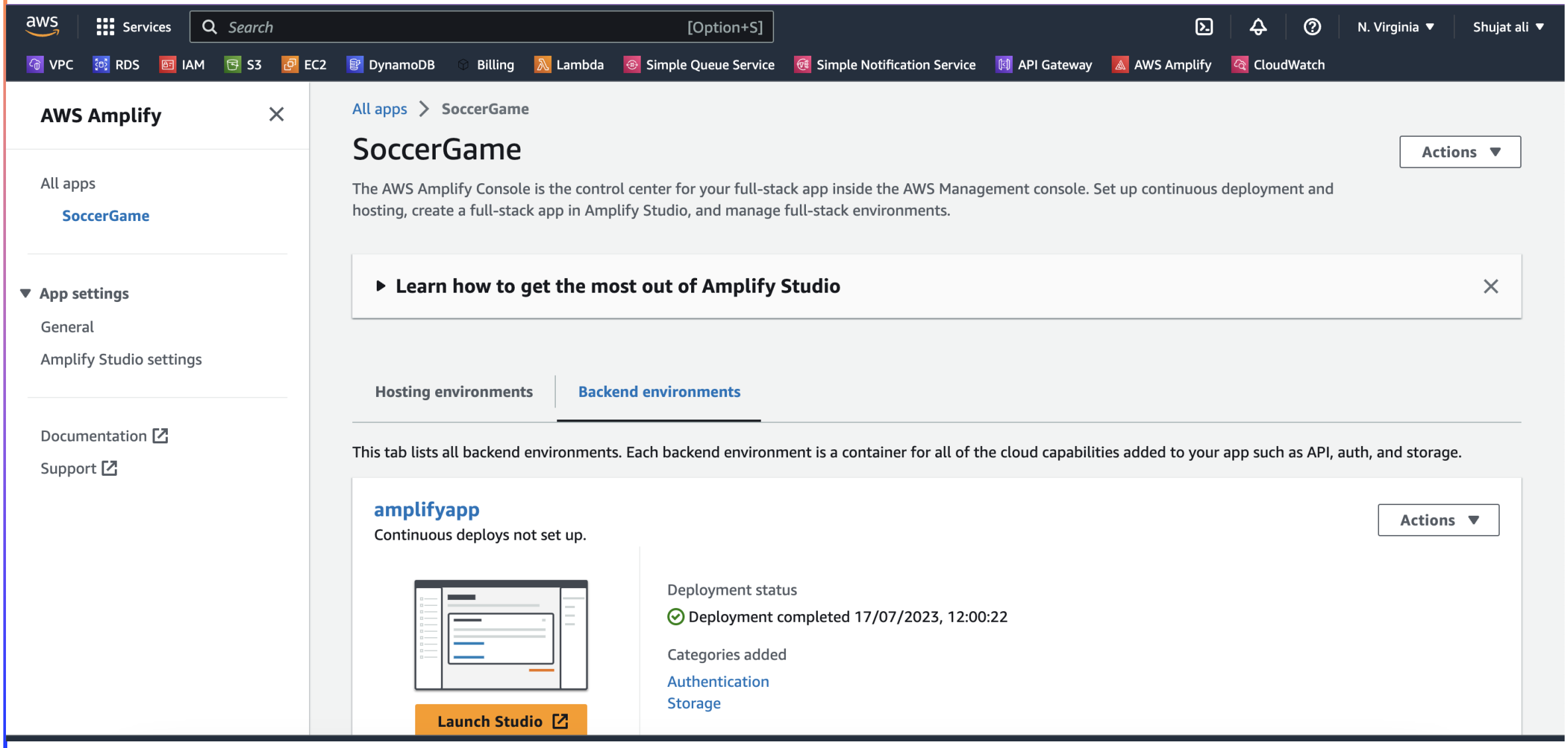
AWS Amplify


aws
Services
Search
[Option+S]
Ohio
Shujat ali
VPC
RDS
IAM
S3
EC2
DynamoDB
Billing
Lambda
Simple Queue Service
Simple Notification Service
API Gateway
AWS Amplify
CloudWatch
AWS Amplify
All apps
Documentation
Support
AWS Amplify
Fastest, easiest way to develop mobile and web apps that scale.
GET STARTED
AWS Amplify is a set of products and tools that enable mobile and front-end web developers to build and deploy secure, scalable full-stack applications, powered by AWS.
AWS Amplify: Fastest, Easiest W...

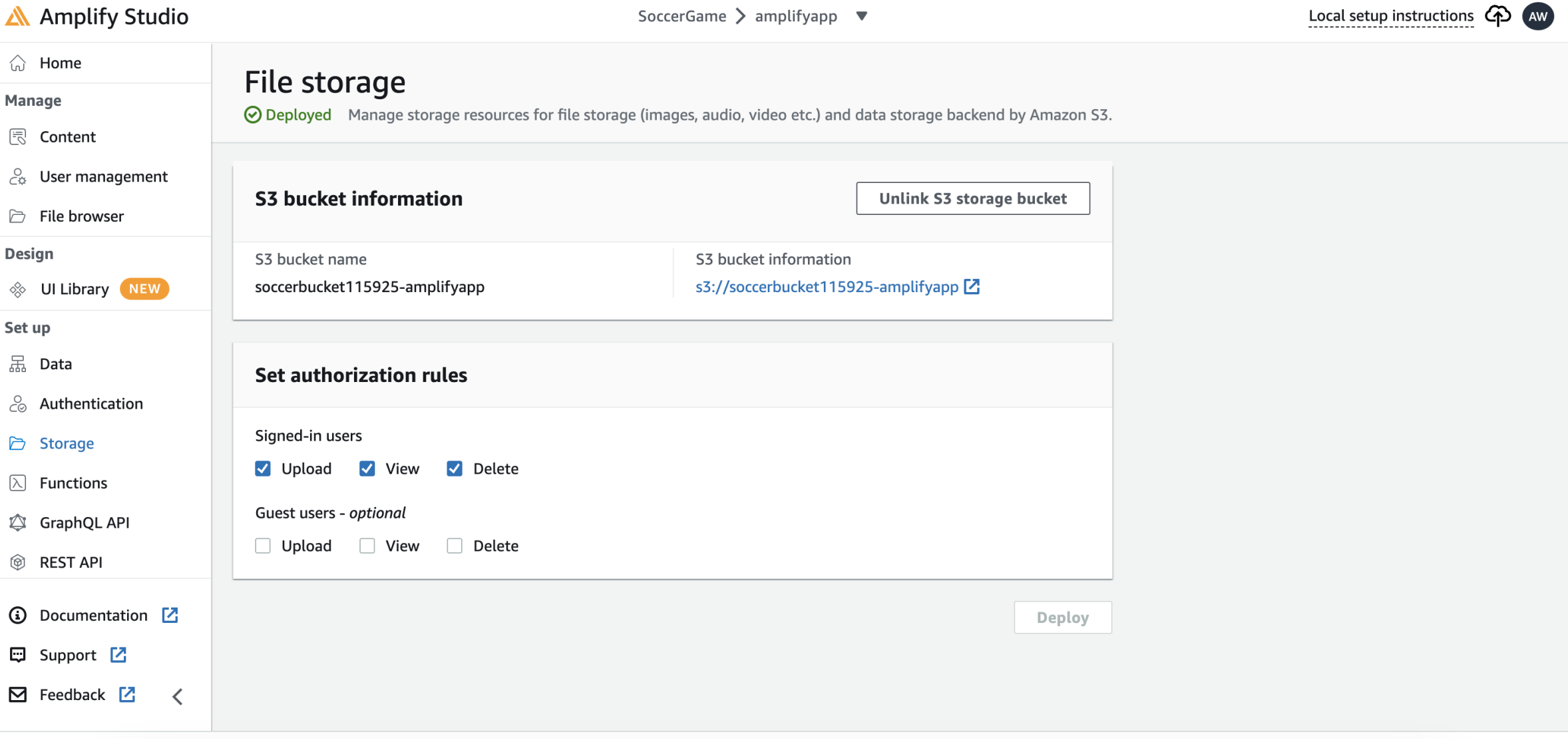
AWS Amplify


Amplify Studio
SoccerGame > amplifyapp
Local setup instructions
AW
Home
Manage
Content
User management
File browser
Design
UI Library
NEW
Set up
Data
Authentication
Storage
Functions
GraphQL API
REST API
Documentation
Support
Feedback
User management
Create, view, and manage users and groups for your application.
Users
Groups
Users
Actions
Create user
Search Username
Find users by Username
1
Login
Created Date
Status
shujat.dev@gmail.com
July 17, 2023 12:05 PM
CONFIRMED

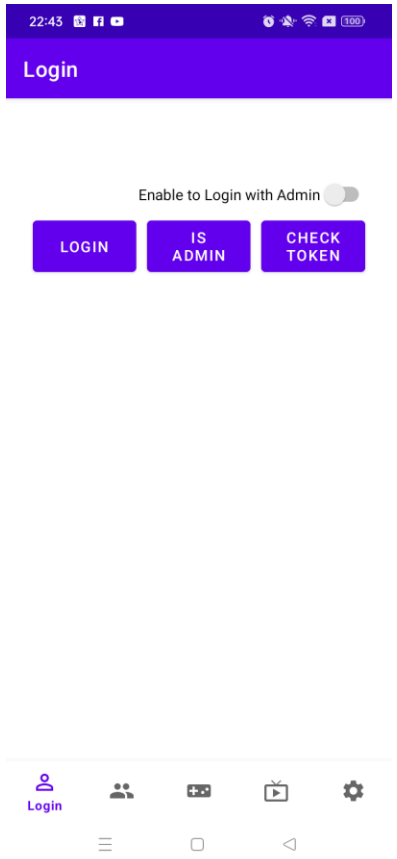
AWS Amplify

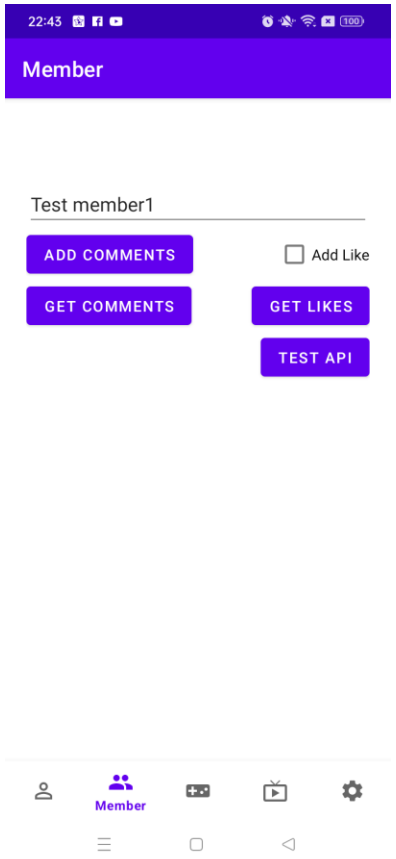

aws
Services
Search
[Option+S]
N. Virginia
Shujat ali
VPC
RDS
IAM
S3
EC2
DynamoDB
Billing
Lambda
Simple Queue Service
Simple Notification Service
API Gateway
AWS Amplify
CloudWatch
AWS Amplify
All apps
SoccerGame
App settings
General
Amplify Studio settings
Documentation
Support
All apps > SoccerGame
SoccerGame
Actions
The AWS Amplify Console is the control center for your full-stack app inside the AWS Management console. Set up continuous deployment and hosting, create a full-stack app in Amplify Studio, and manage full-stack environments.
Learn how to get the most out of Amplify Studio
Hosting environments
Backend environments
This tab lists all backend environments. Each backend environment is a container for all of the cloud capabilities added to your app such as API, auth, and storage.
amplifyapp
Continuous deploys not set up.
Actions
Launch Studio
Deployment status
Deployment completed 17/07/2023, 12:00:22
Categories added
Authentication
Storage

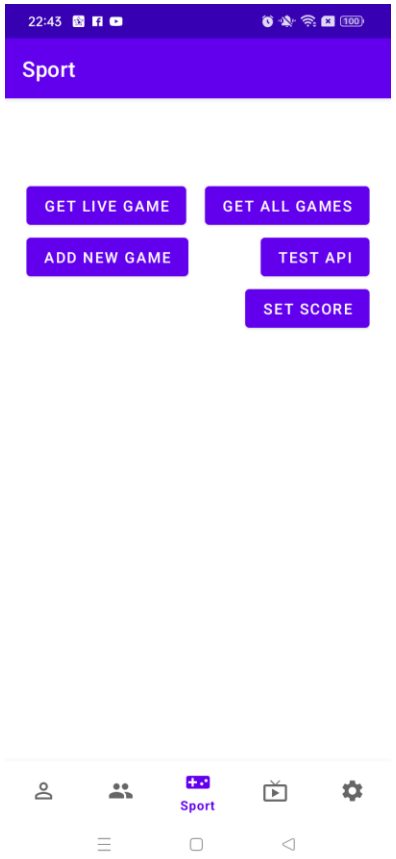
AWS Amplify

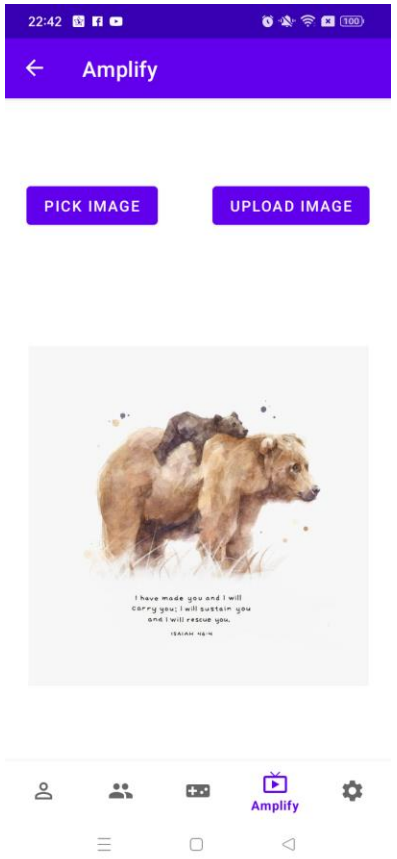

Amplify Studio
SoccerGame > amplifyapp
Local setup instructions
AW
Home
Manage
Content
User management
File browser
Design
UI Library
NEW
Set up
Data
Authentication
Storage
Functions
GraphQL API
REST API
Documentation
Support
Feedback
File storage
Deployed
Manage storage resources for file storage (images, audio, video etc.) and data storage backend by Amazon S3.
S3 bucket information
Unlink S3 storage bucket
S3 bucket name
soccerbucket115925-amplifyapp
S3 bucket information
s3://soccerbucket115925-amplifyapp
Set authorization rules
Signed-in users
Upload
View
Delete
Guest users - optional
Upload
View
Delete
Deploy

Android Mobile Client

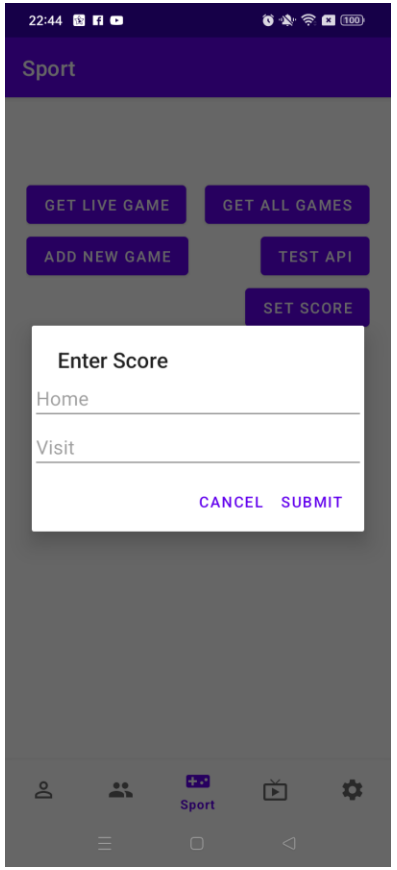


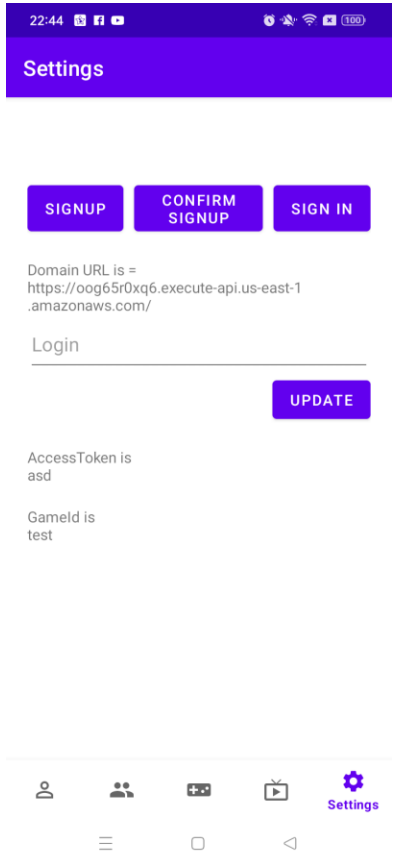


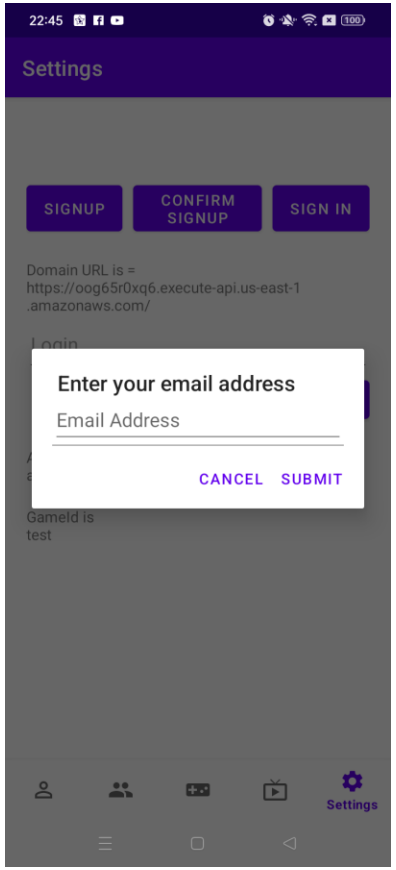


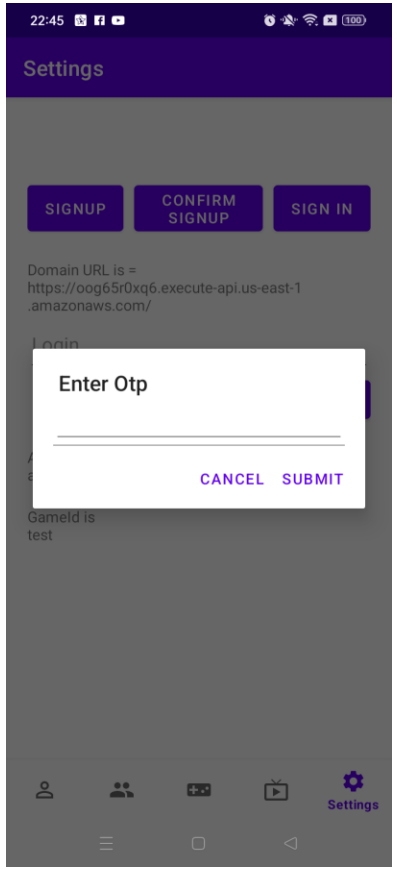


Android Mobile Client









THANK YOU